In the Specification

Please amend the paragraph beginning at line 9 of page 3 as follows:

--The density evolution method is simple for a binary erasure channel. A binary erasure channel is a binary input channel with three output symbols: 0, 1, and an erasure, which can be represented by a question mark "?." Because this method is important background information for the method according to the invention, it is distinguished in greater detail.--

a1

Please amend the paragraph beginning at line 20 of page 5 as follows:

--Similarly, there is a message m_{ai} sent from each check node a to all the variable nodes i connected to the check node. These messages are interpreted as directives from the check node a to the variable node i about what state the variable node should be in. This message is based on the states of the other variable nodes connected to the check node. The ~~check-to-bit~~ check-to-variable messages can, in principle, take on the values 0, 1, or ?, but again only the two messages 0 and ? are relevant when the all-zeros codeword is transmitted.--

a2

Please amend the paragraph beginning at line 4 of page 7 as follows:

--The equation for p_{ia} is

$$p_{ia} = x \prod_{b \in N(i) \backslash a} q_{bi} , \qquad (3)$$

where $b \in N(i) \backslash a$ represents all check nodes directly connected to a neighboring variable node i, except for check node a. This equation can be derived from the fact that for a message m_{ia} to be an erasure, the variable node i must be erased in

a3

transmission, and all incoming messages from other ~~checks~~ check nodes are erasures as well. Of course, if the incoming messages are correlated, then this equation is not correct. However, in a Tanner graph with no loops, each incoming message is independent of all other messages.--

a3

Please amend the paragraph beginning at line 14 of page 7 as follows:

--Similarly, the equation

$$q_{ai} = 1 - \prod_{j \in N(a) \setminus i} (1 - p_{ja}) \qquad (4)$$

can be derived from the fact that a ~~message q_{ai}~~ message m_{ai} can only be in a 0 or 1 state when all incoming messages are in either a zero or one state.--

a4

Please amend the paragraph beginning at line 19 of page 7 as follows:

--The density evolution equations (3) and (4) can be solved by iteration. A good initialization is $p_{ia} = x$ for all messages from variable nodes to check nodes and $q_{ai} = 0$ for all messages from check nodes to variable nodes, as long as the iteration begins with the ~~q_{ai} messages~~ m_{ai} messages. The BEC density evolution equations ultimately converge. This can be guaranteed for codes defined in graphs without loops. It is possible to determine b_i, which is the probability of a failure to decode at variable node i, from the formula

$$b_i = x \prod_{a \in N(i)} q_{ai} .\text{--}$$

a5

Please amend the paragraph beginning at line 1 of page 9 as follows:

--If the focus is on the last bit, then the message is decoded, unless one of the following messages is sent: 00??, 0???, ?0??, ??0? or ????. Therefore, the overall probability that the fourth bit is not decoded is $x^2(1 - x)^2 + 3x^3(1 - x) + x^4 = x^2 + x^3 - x^4$. In the density evolution method, the values for the following variables:

$p_{11}, p_{21}, p_{22}, p_{32}, p_{42}, q_{11}, q_{12}, q_{22}, q_{23}, q_{24}, b_1, b_2, b_3, b_4$

are determined by equations

$$p_{11} = x \quad (7)$$

$$p_{21} = xq_{22} \quad (8)$$

$$p_{22} = xq_{12} \quad (9)$$

$$p_{32} = x \quad (10)$$

$$p_{42} = x\cancel{q_{11} = p_{21}} \quad \cancel{(11)}\underline{(11a)}$$

$$\underline{q_{11} = p_{21}} \quad \underline{(11b)}$$

$$q_{12} = p_{11} \quad (12)$$

$$q_{22} = 1 - (1 - p_{32})(1 - p_{42}) \quad (13)$$

$$q_{23} = 1 - (1 - p_{22})(1 - p_{42}) \quad (14)$$

$$\cancel{q_{24} = 1 - (1 - p_{22})(1 - p_{42})}\ \underline{q_{24} = 1 - ((1 - p_{32})(1 - p_{22}))} \quad (15)$$

and

$$b_1 = xq_{11} \quad (16)$$

$$b_2 = xq_{12}q_{22} \quad (17)$$

$$b_3 = xq_{23} \quad (18)$$

$$b_4 = xq_{24}. \quad (19)\text{--}$$

Please amend the paragraph beginning at line 22 of page 10 as follows:

a7

--If all local neighborhoods in the Tanner graph are identical, the density evolution equations can be simplified. For example, if each variable node i is connected to d_v ~~parity checks~~ check nodes, and each check node a is connected to d_c variable nodes, then all the p_{ia} are equal to the same value p, all the q_{ai} are equal to the same value q, and all the b_i are equal to the same value b. Then,

$$p = xq^{d_v - 1} \tag{34}$$

$$q = 1 - (1 - p)^{d_c - 1} \tag{35}$$

and

$$b = xq^{d_v} \tag{36}$$

which are the density evolution equations for (d_v, d_c) regular Gallager codes, valid in the $N \to \infty$ limit. A regular Gallager code is a sparse random parity check matrix characterized by the restriction that each row has exactly d_c ones in it, and each column contains exactly d_v ones.--

Please amend the paragraph beginning at line 22 of page ~~17~~ 15 as follows:

a8

--The basis of our RG method is the RG transformation by which we iteratively eliminate~~, i.e., "renormalize,"~~ single nodes in the decorated Tanner graph, and adjust the remaining values of the p and q messages so that the smaller error-correcting code has a decoding failure rate as close as possible to the replaced code.--

Please amend the paragraph beginning at line 4 of page 17 as follows:

a9

--Otherwise, if there are no "leaf" ~~check nodes~~ check nodes, renormalize 940 a variable node from among those farthest from the target node that is directly connected to the fewest number of check nodes. Again, ties can be broken randomly.--

Please amend the paragraph beginning at line 1 of page 18 as follows:

a10

--We always initialize our decorated Tanner graph such that all $\cancel{b_I = x}$ $b_i=x$, $p_{ia} = x$, and all $q_{ai} = 0$. We are interested in the decoding failure rate b_i at the specified target variable node i. Our method 900 obtains b_i by repeatedly renormalizing nodes, one node at the time, as described above, other than the target ~~variable~~ node i itself.--

Please amend the paragraph beginning at line 7 of page 18 as follows:

a11

--The first possibility is to ~~renormalize the farthest~~ renormalize a "leaf" variable ~~node~~ node i that is connected to the ~~target node i~~ check node a. Clearly, when that leaf variable node "vanishes," p_{ia} and q_{ai} are also discarded. We also renormalize all the q_{aj} variables leading out of the ~~target node i~~ check node a to other variable nodes j. Our formulation for this renormalization is:

$$q_{aj} \leftarrow 1-(1-q_{aj})(1-p_{ia}) \qquad (37)$$

where the left arrow indicates that we replace the old value of q_{aj} with the new value. Notice that each renormalization of q_{aj} *increases* its value. --

Please amend the paragraph beginning at line 15 of page 18 as follows:

--When we renormalize a "leaf" check node a that is only connected to a single variable node, we adjust the values of all the p_{ib} variables leading to other checks nodes b to which the ~~target node i~~ <u>variable node i</u> is attached. The renormalization group transformation is

$p_{ib} \leftarrow p_{ib} q_{ai}$.--

a12

Please amend the paragraph beginning at line 25 of page 18 as follows:

--The renormalization of step 940 is described in greater detail below for ~~loopy graphs~~ <u>graphs with loops</u>.--

a13

Please amend the paragraph beginning at line 1 of page 19 as follows:

--When only the "target" node i remains, we use the current value of ~~node~~ b_i as our RG prediction of the average failure rate. As stated above, these steps can be repeated for any number of target nodes.--

a14

Please amend the paragraph beginning at line 13 of page 19 as follows:

--We desire to determine the decoding failure rate at the second variable node b_2. We initialize $\cancel{p = p_{21} = p_{22} = p_{32} = p_{42} = x}$ $\underline{p_{11} = p_{21} = p_{22} = p_{32} = p_{42} = x}$, $q_{11} = q_{12} = q_{22} = q_{23} = q_{24} = 0$, and $\cancel{b_2 = 0}$ $\underline{b_1 = b_2 = b_3 = b_4 = x}$.--

a15

Please amend the paragraph beginning at line 19 of page 20 as follows:

a16

--For an error-correcting code represented by a graph with loops, we eventually have to renormalize 940 a variable node that is not a "leaf" node. Note that we never have to renormalize a non-leaf check node. To do this, we first collect all the check nodes a, b, etc. connected to the target node i. We discard q_{ai}, q_{bi}, p_{ia}, p_{ib}, etc. For any given check node attached to variable node i, e.g., check node a, we also collect all the other variable nodes j attached to node a, and renormalize the values of q_{aj}.--

Please amend the paragraph beginning at line 18 of page 22 as follows:

a17

--Converting such a logical argument into an equation for probabilities is straightforward. If we have "m_1 **and** m_2" for two independent messages in a logical argument, then we translates these terms to (p_1 p_2) for the corresponding probabilities, while "m_1 **or** m_2" translates to (1 - (1- p_1) (1 - p_2)). Converting our full logical argument for Figure 4 into an equation for probabilities, enables us to recover an example of the RG transformation of equation (44).--

Please amend the paragraph beginning at line 8 of page 25 as follows:

a18

--For the purposes of describing the exact determination, we instead represent the error-correcting code by a Tanner graph of N nodes, and an associated ~~an erasure~~ erasure probability x_i with each node i of the graph. This "erasure graph" is different than the decorated Tanner graphs 301-305 described above. The decoding failure rate can be determined exactly for an erasure graph, but the exact computation is only practical if the number of nodes in the erasure probability

a18

graph is small. We describe how to determine the decoding failure rate exactly for an erasure probability graph, and then describe how to convert a decorated graph according to the invention into an equivalent erasure graph.--

Please amend the paragraph beginning at line 24 of page 29 as follows:

a19

--The messages m_{ia} are log-likelihood ratios by which the variable node i informs the check node a of its probability of being either a zero or a one. For example, $m_{ia} \to \infty$ means that node i is certain it is a zero, while $m_{ia} = 1$ means that variable node i is signaling check node a that $\ln(p(x_I = 0)/p(x_I = 1)) = 1$. The ~~messages m_{ai}~~ messages m_{ai} are log-likelihood ratios interpreted as information from the check node a to the variable node i about the state of variable node i.

Please amend the paragraph beginning at line 12 of page 30 as follows:

a20

--In the density evolution method for the AWGN channel, one considers the probability distributions $p(m_{ia})$ and ~~$p(m_{ai})$~~ $p(m_{ai})$ for the messages where the probability distribution is an average over all possible received blocks. A distribution $f(x)$ is called consistent if $f(x) = f(-x)e^x$ for all x. The consistency condition is preserved for the message probability distributions for all messages under sum-product decoding.--

Please amend the paragraph beginning at line 19 of page 30 as follows:

a21

--If the probability distributions $p(m_{ia})$ and $p(m_{ai})$ are approximated as Gaussian distributions, then the consistency condition ~~means the means μ~~ means that the means, μ, of these distributions are related to the variances σ^2 by $\sigma^2 = 2\mu$. This

a21 means that the message probability distributions can be characterized by a single parameter: their mean.--

Please amend the paragraph beginning at line 21 of page 31 as follows:

a22 --Just as before, we construct a set of RG transformations which exactly reproduce the density evolution equations for a tree-like graph. We generate a decorated Tanner/Wiberg graph for the code by keeping u_{ai} and v_{ia} variables between each pair of connected nodes. The u_{ai} variables are initialized to infinity, while the v_{ia} variables are initialized to u^0, unless the ith node is a hidden node, in which case the v_{ia} are initialized to zero. We also introduce the ~~variables h_I~~ <u>variables h_i</u> analogous to b_i in the BEC, which are initialized like the v_{ia} variables.--

Please amend the paragraph beginning at line 19 of page 34 as follows:

a23 --On the other hand, determining the bit error rate of every node slows down a search. It may be worthwhile, at least ~~at least~~ for large block-lengths, to restrict oneself to those codes for which there are only a small number of different classes of nodes, defined in terms of the local neighborhoods of the nodes. Most of the best-known codes are of this type. Rather than determining the bit error rate for every variable node, we can determine the bit error rate for just one representative node of each class of variable nodes.--